HISTORICAL RATIONALE

OASIS 1 was designed from the very beginning making use of flagfiles. They allow the user process to tell the cron that there is work to be done. When we started thinking about the design for OASIS 2, one of the first ideas proposed was using a web service for communication, but it was rejected based on security reasons. So I designed OASIS 2.0.x following the same flagfiles model. Moreover, in OASIS 2, the flagfiles are used not only to catch daemon's attention but also to host information. The daemon writes content for the users in the flagfiles. The flagfiles are also used -both OASIS 1 and 2- to lock processes when someone is already publishing.

PROBLEM

There was a problem with the design of OASIS 2. In the case of a process is currently publishing, and during that window 2 new users want to publish, there is no guarantee their arrival order would be respected. That means that both would have to wait for the current one to finish. But after that, the first one noticing it will start. Not necessarily the first one who arrived. FIFO is not enforced.

So I have been working with the code to solve that problem. It is in SVN, in a new branch. To be tested!! During the process of fixing that, I realized the whole flagfile mechanism is quite poor, error prone, and ugly. And the code is not very robust either.

PROBLEM 2

The flagfiles mechanism was always meant to be temporary solution anyways, because they require a shared filesystem between the login host and the repo host, so both processes can read/write them. But, as we want to allow login host to be separate from repo host, even with no shared filesystem, a different mechanism is needed.

PROBLEM 3

There are other pieces of code in OASIS 2 that are not very robust or elegant either. For example, the loops for timeouts.

THINKING ABOUT IT...

So I started thinking that the OASIS architecture should consist on 2 daemons. One for the login host, and one for the repo host. And these 2 daemons can communicate each other, and each one of them should keep track of their own business (either their own history files, or a local DB, etc.) This also has the advantage that the login process does not retain the prompt for the users. Users would talk to this daemon, and prompt is recovered. Just like condor_submit, but oasis-user-publish. Then they can query time to time to see if their job finished. Just like condor_q, but oasis-user-query (or similar). The repo daemon has already one thread per repo. But instead of looking to flagfiles, they would listen to the login daemon, and each one would decide what job to do. And I noticed it is just like a startd with a well defined START expression.

THEREFORE

So I now think OASIS 2 is over-engineered, and with no very good result anyways, as described. However, a very simple an elegant solution has already became clear: using condor. So I proposal re-design OASIS based entirely on condor. The login processes would be thin wrapper around condor_submit/condor_q/condor_history The already existing daemon for the repo host would almost disappear and be replaced by the startd. Things like the timeouts are embedded in the periodic_remove expression in the condor submit file. Serialization comes for free, so no more problems on how to ensure FIFO-behavior in the code.

And extra advantages. The login can be remote. For example, it can be at FNAL. But the server still at the GOC. We would need to figure out the security part. So people can have their login hosts at home, having their own auth/auth mechanisms as they please, but still the difficult part would happen at GOC. And they don't need to to maintain the CVMFS server.

PROPOSAL

So my proposal is to not put into production the OASIS 2 code, and move straight forward to OASIS 3 with a new design based on condor.